

An Efficient Coded Multicasting Scheme Preserving the Multiplicative Caching Gain

Giuseppe Vettigli^{*}, Mingyue Ji[†], Antonia M. Tulino^{*‡}, Jaime Llorca[‡], Paola Festa^{*}

^{*}Università di Napoli Federico II, Napoli, Italy. Email: {vettigli, festa}@unina.it

[†]EE Department, University of Southern California, CA. Email: mingyuej@usc.edu

[‡]Alcatel Lucent, Bell labs, NJ. Email: {a.tulino, jaime.llorca}@alcatel-lucent.com

Abstract—Coded multicasting has been shown to be a promising approach to significantly improve the caching performance of content delivery networks with multiple caches downstream of a common multicast link. However, achievable schemes proposed to date have been shown to achieve the proved order-optimal performance only in the asymptotic regime in which the number of packets per requested item goes to infinity. In this paper, we first extend the asymptotic analysis of the achievable scheme in [1], [2] to the case of heterogeneous cache sizes and demand distributions, providing the best known upper bound on the fundamental limiting performance when the number of packets goes to infinity. We then show that the scheme achieving this upper bound quickly loses its multiplicative caching gain for finite content packetization. To overcome this limitation, we design a novel polynomial-time algorithm based on random greedy graph-coloring that, while keeping the same finite content packetization, recovers a significant part of the multiplicative caching gain. Our results show that the order-optimal coded multicasting schemes proposed to date, while useful in quantifying the fundamental limiting performance, must be properly designed for practical regimes of finite packetization.

I. INTRODUCTION

Recent studies [1]–[10] have been able to characterize the information theoretic limiting performance of several caching networks of practical relevance, in which throughput scales linearly with cache size, showing great promise to accommodate the exponential traffic growth experienced in today's communication networks [11].

Consider a network with one source (server), having access to m files, and n users (caches), each with a storage capacity of M files. In [4], the authors showed that if the users can communicate between each other via Device-to-Device (D2D) communications, a simple distributed random caching scheme and TDMA-based unicast D2D delivery achieves the order-optimal¹ worst-case throughput law $\Theta(\max\{\frac{M}{m}, \frac{1}{m}, \frac{1}{n}\})$, whose linear scaling with M when $Mn \geq m$ exhibits a remarkable multiplicative caching gain. Moreover, in this scheme each user caches entire files without the need of partitioning files into packets, and missed files are delivered via unicast transmissions between neighbor nodes, making it efficiently implementable in practice.

In the case that users cannot communicate between each other, but share a multicast link from the content source, the

authors in [6], [7] showed that the use of coded multicasting allows achieving the same order-optimal worst-case throughput as in the D2D caching network, proving the remarkable fact that multiplicative caching gains can be preserved even if caches cannot communicate between each other. However, the linear coding schemes in [6], [7] involve a number of computations that can grow exponentially with the number of users (caches) and the proved order-optimal performance is only guaranteed when the number of packets per requested file either goes to infinity or also grows exponentially with the number of users.

In [1], the authors considered the same shared link network under random demands characterized by a demand distribution, and proposed a scheme consisting of a random popularity-based (RAP) caching policy and a chromatic-number index coding (CIC) based multicasting scheme, referred to as RAP-CIC, proved to be order-optimal in terms of average throughput. The authors further provided the optimal scaling laws for all regions of the Zipf [12] demand distribution, whose analytical characterization required resorting to a polynomial-time approximation of CIC, referred to as greedy constrained coloring (GCC). While GCC exhibits polynomial complexity in the number of users and packets, the order-optimal performance guarantee still requires the number of packets per file to go to infinity.

It is then key to understand if using any of above referenced schemes, the promising linear throughput scaling with cache size (multiplicative caching gain) can be preserved in practical settings of finite file packetization. In this paper, we address this important open problem focusing on a non-homogenous caching network with a shared multicast link, where users make requests according to possibly different demand distributions and have possibly different cache sizes. The contributions of this paper are as follows. First, we extend RAP-CIC and RAP-GCC to the non-homogenous shared link network and quantify their average performance. Next, we focus on finite file packetization regimes and numerically show that neither GCC nor the coded delivery schemes in [6], [7] can guarantee the promising performance. Finally, we introduce a novel algorithm based on a greedy randomized approach referred to as Greedy Randomized Algorithm Search Procedure (GRASP), which is shown to recover a significant part of the multiplicative caching gain in the same regimes of finite file packetization, while incurring a complexity at most

¹Order-optimal means that the gap between the information theoretic converse and the achievable throughput can be bounded by a constant number when $m, n \rightarrow \infty$.

quadratic in the number of requested packets.

II. NETWORK MODEL AND PROBLEM FORMULATION

We consider a network consisting of a source node with access to a content library of m files $\mathcal{F} = \{1, \dots, m\}$ each of size F bits, and n user nodes $\mathcal{U} = \{1, \dots, n\}$. The source node communicates with the user nodes through a shared multicast link of finite capacity C . Without loss of generality, we assume $C = F$ bits/unit time and measure transmission rate in time units or file-size transmissions necessary to deliver the requested files to the users. Each user $u \in \mathcal{U}$ has a cache of size $M_u F$ bits (*i.e.*, M_u files). The channel between the source and all the users follows a shared error-free deterministic model. User u requests file f with probability $q_{f,u}$, where $q_{f,u} \in [0, 1]$ and $\sum_{f=1}^m q_{f,u} = 1$. We denote by $\mathbf{Q} = [q_{f,u}]$, $u = 1, \dots, n$, $f = 1, \dots, m$, the demand (or content popularity) distribution, and by $\mathbf{f} = \{f_1, \dots, f_n\}$ the request vector, where f_u is the file requested by user u . The goal is to design a content distribution scheme consisting of a caching placement (configuration of the user caches) and a delivery scheme (multicast codeword to be sent to all users) such that all demands are satisfied with probability 1 and the expected rate $\bar{R}(\mathbf{Q})$ is minimized. The expectation is over the demand distribution \mathbf{Q} . We denote the minimum achievable expected rate by $\bar{R}^*(\mathbf{Q})$.

III. GENERAL ACHIEVABLE SCHEME

In this section, we extend the analysis of RAP-CIC [1], an achievable scheme based on random popularity-based (RAP) caching and chromatic index coding (CIC) delivery, to the heterogeneous shared link network introduced in Section II.

A. Random Popularity-based (RAP) Caching

As in [1], [2], let each file be partitioned into B equal-size packets, represented as symbols of $\mathbb{F}_{2^{F/B}}$ for $F/B \rightarrow \text{constant}$ as $F, B \rightarrow \infty$. We denote by \mathbf{C} and \mathbf{W} the realizations of the *packet level* caching and demand configurations, respectively, where $\mathbf{C}_{u,f}$ denotes the packets of file f cached by user u , and $\mathbf{W}_{u,f}$ the packets of file f requested by user u . We use the caching algorithm in Fig. 1 to let each user fill its own cache independently by knowing the caching distribution $\mathbf{P} = [p_{f,u}]$ with $0 \leq M_u p_{f,u} \leq 1, \forall f$, and $\sum_{f=1}^m p_{f,u} = 1, \forall u$. In line with [1], we refer to the caching policy in Fig. 1 that uses the caching distribution that minimizes the upper bound of the optimal expected rate given in Section III-C as RAP.

B. Chromatic Index Coding (CIC) Delivery

The CIC delivery scheme is based on a minimum vertex coloring of the corresponding index coding conflict graph [13], $\mathcal{H}_{\mathbf{C}, \mathbf{W}} = (\mathcal{V}, \mathcal{E})$, constructed as follows:

- Each packet requested by each user is represented as a distinct vertex in \mathcal{V} . Each vertex $v \in \mathcal{V}$ is hence uniquely identified by a pair $\{\rho(v), \mu(v)\}$ where $\rho(v)$ indicates the packet identity and $\mu(v)$ the user requesting it.
- For any two vertices v_1, v_2 , we say that vertex v_1 interferes with vertex v_2 if the packet associated with

```

algorithm Caching algorithm (P)
1 for  $f \in \mathcal{F}$ 
2   Each user  $u \in \mathcal{U}$  caches a subset  $\mathbf{C}_{u,f}$  of
    $p_{f,u} M_u B$  distinct packets of file  $f$ 
   uniformly at random;
3 endfor
4  $\mathbf{C} \leftarrow \{\mathbf{C}_{u,f}, u = 1, \dots, n, f = 1, \dots, m\}$ ;
5 return( $\mathbf{C}$ );
end Caching algorithm

```

Fig. 1. The random caching algorithm.

$v_1, \rho(v_1)$, is not in the cache of the user associated with $v_2, \mu(v_2)$, and $\rho(v_1)$ and $\rho(v_2)$ do not represent the same packet. There exists an undirected edge between v_1 and v_2 if v_1 interferes with v_2 or v_2 interferes with v_1 .

Given a minimum vertex coloring of the conflict graph $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$, the corresponding index coding scheme transmits the modulo sum of the packets (vertices in $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$) with the same color. Therefore, given \mathbf{C} and \mathbf{W} , the total number of packet transmissions given by the chromatic number of the conflict graph $\chi(\mathcal{H}_{\mathbf{C}, \mathbf{W}})$, which yields a transmission rate of $\chi(\mathcal{H}_{\mathbf{C}, \mathbf{W}})/B$.

C. Achievable Expected Rate

Given the caching and demand distributions \mathbf{P} and \mathbf{Q} , the asymptotic expected rate is computed as the expected chromatic number of the conflict graph when the number of packets goes to infinity, $\bar{R}(\mathbf{P}, \mathbf{Q}) \triangleq \lim_{B \rightarrow \infty} \mathbb{E}[\chi(\mathcal{H}_{\mathbf{C}, \mathbf{W}})/B | \mathbf{C}]$,² where the expectation is taken only over the demand distribution \mathbf{Q} . The expected rate $\bar{R}(\mathbf{P}, \mathbf{Q})$ is hence a random variable, which is a function of the random caching configuration \mathbf{C} . We now upper bound $\bar{R}(\mathbf{P}, \mathbf{Q})$ using the following Theorem:

Theorem 1: For any given m, n, M , and \mathbf{Q} , when $B \rightarrow \infty$, the expected rate $\bar{R}(\mathbf{P}, \mathbf{Q})$ achieved by a content distribution scheme that uses caching policy in Fig. 1 with caching distribution \mathbf{P} and CIC delivery, satisfies

$$\bar{R}(\mathbf{P}, \mathbf{Q}) \leq \bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q}) \triangleq \min\{\psi(\mathbf{P}, \mathbf{Q}), \bar{m}\}, \quad (1)$$

with high probability.³ In (1),

$$\bar{m} = \sum_{f=1}^m \left(1 - \prod_{u=1}^n (1 - q_{f,u}) \right), \quad (2)$$

and

$$\psi(\mathbf{P}, \mathbf{Q}) = \sum_{\ell=1}^n \sum_{\mathcal{U}^\ell \subset \mathcal{U}} \sum_{f=1}^m \sum_{u \in \mathcal{U}^\ell} \rho_{f,u, \mathcal{U}^\ell} \lambda(u, f), \quad (3)$$

where

$$\lambda(u, f_u) = (1 - p_{f_u, u} M_u) \prod_{k \in \mathcal{U}^\ell \setminus \{u\}} (p_{f_u, k} M_k) \prod_{k \in \mathcal{U} \setminus \mathcal{U}^\ell} (1 - p_{f_u, k} M_k),$$

² $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$ denotes the random conflict graph, which is a function of the random caching and demand configurations, \mathbf{C} and \mathbf{W} , respectively.

³The term "with high probability" means that $\lim_{B \rightarrow \infty} \mathbb{P}(\bar{R}(\mathbf{P}, \mathbf{Q}) \leq \bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})) = 1$.

\mathcal{U}^ℓ denotes a set of users with cardinality ℓ , and

$$\rho_{f,u,\mathcal{U}^\ell} \triangleq \mathbb{P}(f = \arg \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \lambda(u, f_u)),$$

denotes the probability that f is the file that maximizes the term $\lambda(u, f)$ among $\mathbf{f}(\mathcal{U}^\ell)$, the set of files requested by the subset of ℓ users \mathcal{U}^ℓ . \square

We remark that under homogeneous content popularity and cache size, *i.e.*, when $q_{f,u} = q_f, M_u = M, \forall u \in \mathcal{U}$, then $p_{f,u} = p_f, \forall u \in \mathcal{U}$, and the generalized upper bound of Theorem 1 becomes the order-optimal expected rate for homogenous shared link networks proved in [1].

The analytical characterization of the achievable expected rate given by Theorem 1 can then be used to obtain the desired caching distribution for a wide class of heterogeneous network models. In particular, we denote by \mathbf{P}^* the caching distribution that minimizes $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$, and refer to the scheme that uses caching algorithm in Fig. 1 with $\mathbf{P} = \mathbf{P}^*$ and CIC delivery as RAP-CIC.

IV. POLYNOMIAL-TIME ALGORITHMS

As described in Section III-B, CIC delivery involves computing a minimum vertex coloring of the conflict graph $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$. The graph coloring problem is a well known NP-complete problem [14]; indeed, given an undirected graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, Garey and Johnson showed that obtaining colorings using $s \cdot \chi(\mathcal{H})$ colors, where $s < 2$, is NP-hard [15].

In this section, we describe two polynomial-time delivery schemes. The first one is based on a greedy constrained coloring (GCC) algorithm introduced in [1]. The authors in [1] proved that the rate of RAP-GCC converges, as the number of packets $B \rightarrow \infty$ and for homogeneous shared link networks, to the order-optimal expected rate given in Theorem 1 when $q_{f,u} = q_f, M_u = M, \forall u \in \mathcal{U}$. Following a similar approach, it is immediate to prove that the performance of RAP-GCC in heterogeneous shared link networks converges to the upper bound given in (1) as $B \rightarrow \infty$. It is also easy to verify that RAP-GCC achieves the same performance as the algorithm given in [7] for the worst-case demand setting. We recap GCC in Section IV-A.

The second one is presented in Section IV-B and represents a novel coded multicasting scheme based on a Greedy Randomized Algorithm Search Procedure (GRASP) that exhibits lower polynomial-time complexity than GCC. In Section V, we show that for practical regimes of finite file packetization, while GCC loses the multiplicative caching gain, GRASP is still able to approach the fundamental limiting performance and recover a significant part of the multiplicative caching gain.

A. GCC (Greedy Constrained Coloring)

The GCC algorithm works by computing two valid colorings of the conflict graph $\mathcal{H}_{\mathbf{C}, \mathbf{W}} = (\mathcal{V}, \mathcal{E})$, referred to as GCC_1 and GCC_2 . GCC compares the rate achieved by the

algorithm $\text{GCC}_1(\mathcal{V}, \mathcal{E})$

```

1  Let  $\hat{\mathcal{V}} = \mathcal{V}$ ;
2  Let  $\mathcal{C} = \emptyset$ ;
3   $\mathbf{c}_1 = \emptyset$ ;
4  while  $\hat{\mathcal{V}} \neq \emptyset$ 
5    Pick an arbitrary vertex  $v$  in  $\hat{\mathcal{V}}$ . Let  $\mathcal{I} = \{v\}$ ;
6    for all  $v' \in \hat{\mathcal{V}} \setminus \mathcal{I}$ 
7      if (There is no edge between  $v'$  and  $\mathcal{I}$ )
8         $\cap \{\mathcal{K}_{v'} \equiv \mathcal{K}_{\tilde{v}} : \forall \tilde{v} \in \mathcal{I}\}$  then
9           $\mathcal{I} = \mathcal{I} \cup v'$ ;
10       endif
11    endfor
12    Color all the vertices in  $\mathcal{I}$  by  $c \notin \mathcal{C}$ ;
13    Let  $\mathbf{c}_1[\mathcal{I}] = c$ ;
14     $\hat{\mathcal{V}} = \hat{\mathcal{V}} \setminus \mathcal{I}$ ;
15 endwhile
16 return( $\mathbf{c}_1$ );
end  $\text{GCC}_1$ 
```

Fig. 2. The greedy constrained coloring algorithm GCC_1 . \mathcal{K}_v denotes the set of users that are either caching or requesting packet v .

two coloring solutions and constructs the transmission code based on the coloring with minimum rate.⁴

GCC_1 computes a coloring of the conflict graph $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$ as described in Fig. 2. Note that both the outer while-loop starting at line 4 and the inner for-loop starting at line 6 iterate at most $|\mathcal{V}|$ times. The operation in line 7 has complexity $O(n)$. Therefore, the complexity of GCC_1 is $O(n|\mathcal{V}|^2)$ or equivalently $O(n^3 B^2)$ since $|\mathcal{V}| \leq nB$, which is polynomial in $n, |\mathcal{V}|$ (or equivalently in n, B).

On the other hand, GCC_2 computes a minimum coloring of $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$ subject to the constraint that only the vertices representing the same packet are allowed to have the same color. In this case, the total number of colors is equal to the number of distinct requested packets, and the coloring can be found in $O(|\mathcal{V}|^2)$. It is immediate to see that this scheme corresponds to the naive (uncoded) multicasting transmission of all requested packets.

It can be shown that RAP-GCC achieves the upper bound of the average rate for heterogeneous shared link networks given by (1) (when $B \rightarrow \infty$). However, as will be shown in Section V, for finite B , RAP-GCC loses the promising multiplicative caching gain.

B. Greedy Randomized Algorithm Search Procedure (GRASP)

In this section, we design an efficient metaheuristic to find suboptimal good solutions to the graph coloring problem given in Section III-B in reasonable running times (lower than GCC), and that allow preserving the multiplicative caching gain.

Specifically, we propose a GRASP [16]–[20], whose general framework is described in the following:

- 1) A GRASP performs a certain number of iterations, until a stopping criterion is met (such as, for example, a predefined maximum number of iterations).

⁴Recall that the transmission code (index code) is constructed by the modulo sum of the vertices (packets) in $\mathcal{H}_{\mathbf{C}, \mathbf{W}}$ with the same color.

```

algorithm GRASP_GraphColoring
(MaxIterations,  $\mathcal{V}$ ,  $\mathcal{E}$ ,  $d$ ,  $Adj(\cdot)$ ,  $f(\cdot)$ )
1  $\mathbf{c}_{\text{best}} := \emptyset$ ;  $f(\mathbf{c}_{\text{best}}) := +\infty$ ;
2  $\hat{\mathcal{V}} := \text{sort}(\mathcal{V})$ ;
3 for  $k = 1$  to MaxIterations  $\rightarrow$ 
4    $\mathcal{C} := \emptyset$ ;
5    $\beta := \text{random}[0, 1]$ ;
6    $\mathbf{c} := \text{BuildGreedyRandAdaptive}(\beta, \hat{\mathcal{V}}, \mathcal{E}, d, Adj(\cdot), f(\cdot), \mathcal{C})$ ;
7    $\mathbf{c}^* := \text{LocalSearch}(\hat{\mathcal{V}}, \mathcal{E}, \mathbf{c}, f(\mathbf{c}), \mathcal{C})$ ;
8   if ( $f(\mathbf{c}^*) < f(\mathbf{c}_{\text{best}})$ ) then
9      $\mathbf{c}_{\text{best}} := \mathbf{c}^*$ ;
10     $f(\mathbf{c}_{\text{best}}) := f(\mathbf{c}^*)$ ;
11  endif
12 endfor
13 return( $\mathbf{c}_{\text{best}}$ );
end GRASP_GraphColoring

```

Fig. 3. Pseudo-code of a GRASP for the Graph Coloring Problem. $Adj(i) = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}, \forall i \in \mathcal{V}$.

- 2) At each GRASP iteration,
 - a) a greedy-randomized adaptive solution \mathbf{c} is built;
 - b) starting from \mathbf{c} as initial solution, a local search phase is performed returning a locally optimal solution \mathbf{c}^* ;
- 3) At the end of all GRASP iterations, the best locally optimal solution \mathbf{c}_{best} , *i.e.*, the solution corresponding to the best objective function value $f(\mathbf{c}_{\text{best}})$ is returned as final solution and the algorithm stops.

Our GRASP differs from the GRASP proposed by Laguna and Martí [21] in two main aspects: 1) it is able to handle problem instances characterized by any graph topology, density/sparsity, and size; 2) the local search strategy checks for redundant colors focusing on each vertex, one at the time; while Laguna and Martí's GRASP iteratively merges the colors of a pair of independent sets and focuses only on *illegal vertices* (*i.e.*, those vertices that, after the merge, result colored with the same color as one of their adjacent vertices).

Fig. 3 depicts the pseudo-code of our GRASP heuristic for the Graph Coloring Problem. In the following, we describe the solution construction and the local search procedures performed by our algorithm.

Solution Construction Procedure. Let \mathcal{Q} be a set of $|\mathcal{V}|$ candidate colors. The construction phase assigns to each vertex $i \in \mathcal{V}$ a color $c \in \mathcal{Q}$ in such a way that c is not assigned to any vertex adjacent to i and that the total number of used colors is as smaller as possible. Fig. 4 shows the pseudo-code of the construction procedure. To build a feasible solution starting from an empty solution (line 1), the GRASP construction phase goes through $|\mathcal{V}|$ iterations in a for-loop (lines 2–11), in which colors are assigned to uncolored vertices in a greedy, randomized, and adaptive manner. In particular, at each iteration, the choice of the next vertex to be colored is determined by ordering all currently uncolored vertices in a candidate list $\mathcal{W} = \mathcal{V} \setminus \mathbf{c}$ with respect to a greedy

```

function BuildGreedyRandAdaptive( $\beta, \hat{\mathcal{V}}, \mathcal{E}, d, Adj(\cdot), f(\cdot), \mathcal{C}$ )
1  $\mathbf{c} := \emptyset$ ;
2 for  $j = 1$  to  $|\mathcal{V}| \rightarrow$ 
3    $RCL := \text{MakeRCL}(\beta, \mathcal{V}, \mathcal{E}, d, \mathbf{c})$ ;
4    $i := \text{SelectIndex}(RCL)$ ;
5    $c := \text{GetColor}(\mathcal{V}, \mathcal{E}, i, \mathcal{C}, Adj(\cdot), \mathbf{c})$ ;
6    $\mathbf{c}[i] := c$ ;
7   if ( $c \notin \mathcal{C}$ ) then
8      $\mathcal{C} := \mathcal{C} \cup \{c\}$ ;
9      $f(\mathbf{c}) := |\mathcal{C}|$ ;
10  endif
11 endfor
12 return( $\mathbf{c}$ );
end BuildGreedyRandAdaptive

```

Fig. 4. Pseudo-code of the GRASP construction procedure for the Graph Coloring Problem.

function $g : \mathcal{W} \mapsto \mathbb{R}$ that measures the myopic benefit of selecting each vertex and that in our case is related to the degree of a candidate vertex. The construction is adaptive because the benefit associated with each candidate vertex is updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous vertex. The probabilistic component arises from randomly choosing one of the best candidates in the list \mathcal{W} , but not necessarily the top candidate. The list of best candidates is called the *Restricted Candidates List* (*RCL*) and is built according to Fig. 5. Once the *RCL* is built (line 3), a randomly selected candidate from the *RCL* (line 4) is assigned a color based on the colors already assigned to its adjacent vertices (line 5).

The construction phase hence performs the following steps (as shown in Figs. 4, 5, and 6):

Let $d(i) = |Adj(i)|$, for all $i \in \mathcal{V}$, be the degree of vertex i . Let $\mathbf{c} = \emptyset$ be the solution under construction (initially empty), *i.e.*, the set of vertices already assigned to a color, and let $\mathcal{C} = \emptyset$ (initially empty) be the set of colors that are associated to at least a vertex in \mathbf{c} . At each iteration, the following quantities are computed and operations performed:

- 1) g_{\min} , the minimum greedy value:

$$g_{\min} = \min_{i \in \mathcal{V} \setminus \mathbf{c}} d(i);$$

- 2) g_{\max} , the maximum greedy value:

$$g_{\max} = \max_{i \in \mathcal{V} \setminus \mathbf{c}} d(i);$$

- 3) A threshold value τ :

$$\tau = g_{\min} + [\beta \cdot (g_{\max} - g_{\min})], \text{ where } \beta \in [0, 1];$$

- 4) The *RCL*, as the subset of candidate uncolored vertices whose degree is at least τ :

$$RCL = \{i \in \mathcal{V} \setminus \mathbf{c} \mid d(i) \geq \tau\};$$

- 5) A vertex i is randomly selected from the *RCL* ($i = \text{SelectIndex}(RCL)$ in Fig. 4).

```

function MakeRCL ( $\beta, \mathcal{V}, \mathcal{E}, d, \mathbf{c}$ )
1   $g_{\min} := \min_{i \in \mathcal{V} \setminus \mathbf{c}} d(i);$ 
2   $g_{\max} := \max_{i \in \mathcal{V} \setminus \mathbf{c}} d(i);$ 
3   $\tau := g_{\min} + [\beta \cdot (g_{\max} - g_{\min})];$ 
4   $RCL := \{i \in \mathcal{V} \setminus \mathbf{c} \mid d(i) \geq \tau\};$ 
5  return( $RCL$ );
end MakeRCL

```

Fig. 5. Pseudo-code of the function that builds the RCL at each GRASP construction iteration.

```

function GetColor ( $\mathcal{V}, \mathcal{E}, i, \mathcal{C}, Adj(\cdot), \mathbf{c}$ )
1   $\mathcal{L} := \emptyset;$ 
2  for each  $j \in Adj(i)$   $\mathcal{L} := \mathcal{L} \cup \{\mathbf{c}[j]\};$ 
3  if  $(\mathcal{C} \setminus \mathcal{L} \neq \emptyset)$  then
4     $\mathbf{c}' := \text{SelectColor}(\mathcal{C} \setminus \mathcal{L});$ 
5  else
6     $\mathbf{c}' := \text{NewColor}(\mathcal{C});$ 
7  endif
8  return( $\mathbf{c}'$ );
end GetColor

```

Fig. 6. Pseudo-code of the function that is invoked at each GRASP construction iteration to get an available and feasible color.

Note that the value of $\beta \in [0, 1]$ determines the percentage of greediness versus randomness in the choice of the vertices to be inserted in the RCL at each iteration. In fact, for $\beta = 1$, the choice is totally greedy and only vertices with degree g_{\max} are inserted. On the contrary, for $\beta = 0$, the choice is totally random and all candidate vertices are inserted (i.e., $RCL = \mathcal{V}$);

- 6) Once vertex i is selected, its adjacent vertices are analyzed and the four possible scenarios that may occur are the following:
 - i. All adjacent vertices are still uncolored and the set $\mathcal{C} = \emptyset$: in this case, a new color c is assigned to vertex i and $\mathcal{C} = \mathcal{C} \cup \{c\}$;
 - ii. All adjacent vertices are still uncolored and the set $\mathcal{C} \neq \emptyset$: in this case, vertex i is colored with the first color $c \in \mathcal{C}$ available;
 - iii. At least one adjacent vertex is colored with a color $c \in \mathcal{C}$ and all currently used colors $c \in \mathcal{C}$ are already assigned to at least an adjacent vertex: in this case, vertex i is colored with a new color c' and $\mathcal{C} = \mathcal{C} \cup \{c'\}$;
 - iv. At least one adjacent vertex is colored with a color $c \in \mathcal{C}$ and there is a color $c' \in \mathcal{C}$ that has not been assigned to any adjacent vertex: in this case, vertex i is colored with color c' ;
- 7) Vertex i is inserted into the solution under construction ($\mathbf{c}[i] = c'$ or $\mathbf{c}[i] = c$, according to scenarios 6.i.–6.iv.) and the objective function value is updated (i.e., $f(\mathbf{c}) = |\mathcal{C}|$).

Local Search Procedure. Solutions generated by the GRASP construction are not guaranteed to be locally optimal

with respect to simple neighborhood definitions. It is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the current neighborhood. In our GRASP, the local search algorithm has the purpose of checking redundancy of each color $c \in \mathcal{C}$, in order to eventually decrease the current objective function value $|\mathcal{C}|$.

In more detail, the local search computes, iteratively for each color $c \in \mathcal{C}$, the set \mathcal{G}_c of all vertices colored with color c and performs the following steps:

- 1) For each vertex $i \in \mathcal{G}_c$, span $Adj(i)$: if there is a color $c' \in \mathcal{C}$, $c' \neq c$, not assigned to any adjacent vertex $j \in Adj(i)$, then color vertex i with color c' ;
- 2) Color c is removed from the set \mathcal{C} if and only if in Step 1 it has been possible to replace c with some color $c' \neq c$.

Computational complexity of the proposed GRASP. The complexity analysis of the proposed GRASP algorithm boils down to the following steps:

- 1) Sorting the set \mathcal{V} according to a non-ascending order of the degree of the vertices: $O(|\mathcal{V}| \log |\mathcal{V}|)$.
- 2) Solution construction procedure: $O(|\mathcal{E}|)$.
- 3) Local Search Procedure: $O(|\mathcal{E}|)$.

It can be seen that step 1) is performed only once, at the beginning of the algorithm. Since step 2) and step 3) are performed a fixed number of iterations (MaxIterations), it results that the overall computational complexity of the proposed GRASP has computational complexity

$$O(|\mathcal{V}| \log |\mathcal{V}| + \text{MaxIterations} \cdot |\mathcal{E}|). \quad (4)$$

Remark: Observe that the complexity of GRASP is $O(|\mathcal{V}|^2)$, which is a factor of n lower than the complexity ($O(n|\mathcal{V}|^2)$) of GCC.

V. SIMULATIONS AND DISCUSSIONS

In this section, we numerically analyze the performance of the two polynomial-time achievable schemes illustrated in Section IV for finite file packetization. Specifically, assuming the random popularity-based (RAP) caching policy in Fig. 1 with caching distribution $\mathbf{P} = \mathbf{P}^*$ (recall that \mathbf{P}^* is the caching distribution that minimizes $\bar{R}^{\text{ub}}(\mathbf{P}, \mathbf{Q})$ in (1) among all \mathbf{P}), we compare the average performance of GCC and GRASP when files are partitioned into a finite number of packets B . For comparison, we also plot 1) the performance of LFU (Least Frequently Used),⁵ shown to be optimal in uncoded networks, and 2) the performance of GCC for infinite file packetization ($B \rightarrow \infty$), as given in Theorem 1.

For simplicity and to illustrate the effectiveness of GRASP, we consider a homogeneous network scenario, in which users request files according to a Zipf demand distribution with parameter $\alpha \in \{0.2, 0.6\}$ and all caches have size M files.

⁵LFU discards the least frequently requested file upon the arrival of a new file to a full cache of size M_u files. In the long run, this is equivalent to caching the M_u most popular files.

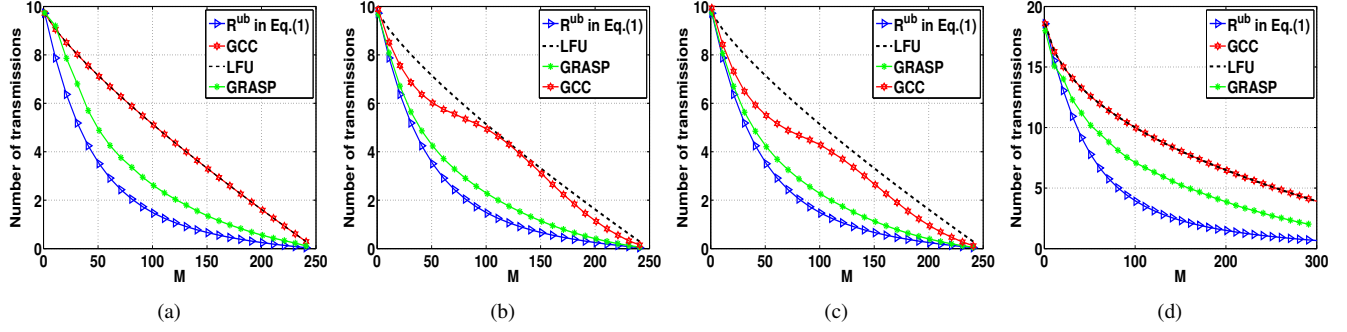


Fig. 7. Average number of transmissions over the shared multicast link. a) $n = 10$, $m = 250$, $B = 20$, and $\alpha = 0.2$; b) $n = 10$, $m = 250$, $B = 100$, and $\alpha = 0.2$; c) $n = 10$, $m = 250$, $B = 200$, and $\alpha = 0.2$; d) $n = 20$, $m = 500$, $B = 50$, and $\alpha = 0.6$.

Further, we assume that when using GCC or GRASP, the source node pre-evaluates the performance of LFU and chooses the minimum of the two accordingly. Hence, denoting by R_{LFU} , R_{GCC} and R_{GRASP} the average rate achieved by LFU, GCC and GRASP, respectively, Fig. 7 plots the performance of GCC and GRASP as $\min\{R_{LFU}, R_{GCC}\}$, and $\min\{R_{LFU}, R_{GRASP}\}$, respectively.⁶

Figs. 7(a), (b), and (c) plot the average rate for a network with $n = 10$ users, $m = 250$ files and Zipf parameter $\alpha = 0.2$. Observe how the significant caching gains (with respect to LFU) quantified by the order-optimal upper bound are completely lost when using GCC with finite packetization $B = 20$, and only slightly recovered as the packetization increases to $B = 100$ and $B = 200$. On the other hand, observe how GRASP remarkably preserves most of the promising multiplicative caching gains for the same values of file packetization. For example, in Fig. 7(b), if M doubles from $M = 50$ to $M = 100$, then the rate achieved by GRASP essentially halves from 4.2 to 2.2. For the same regime, it is straightforward to verify that neither GCC nor LFU exhibits this property. Fig. 7(d) illustrates a scenario with higher popularity skewness, e.g., $\alpha = 0.6$. Observe how, also in this setting, a finite number of packets ($B = 50$) completely limits the gains of GCC. On the other hand, GRASP is still able to preserve significant gains. For example, when M doubles from $M = 70$ to $M = 140$, the achievable rate by GRASP goes from 8.8 to 5.5, approaching a half rate reduction even with only 50 packets per file. Finally note from Fig. 7(b), that in order to guarantee a rate $R = 4$, GCC requires a cache size of $M = 120$, while GRASP can reduce the cache size requirement to $M = 50$, a $2.4\times$ cache size reduction.

REFERENCES

- [1] M. Ji, A.M. Tulino, J. Llorca, and G. Caire, "On the average performance of caching and coded multicasting with random demands," in *ISWCS*, 2014. IEEE, 2014, pp. 922–926.
- [2] M. Ji, A.M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *arXiv preprint:1502.03124*, 2015.
- [3] J. Llorca, A.M. Tulino, K. Guan, and D. Kilper, "Network-coded caching-aided multicast for efficient content delivery," in *ICC*, 2013 *Proceedings*. IEEE, 2013.
- [4] M. Ji, G. Caire, and A.F. Molisch, "The throughput-outage tradeoff of wireless one-hop caching networks," *arXiv:1302.2168*, 2013.
- [5] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *arXiv:1305.5216*, 2013.
- [6] M.A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *Information Theory, IEEE Transactions on*, vol. 60, no. 5, pp. 2856–2867, May 2014.
- [7] M.A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.
- [8] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *arXiv:1405.5336*, 2014.
- [9] M. Ji, A.M. Tulino, J. Llorca, and G. Caire, "Caching and coded multicasting: Multiple groupcast index coding," in *GlobalSIP*, 2014. IEEE, 2014, pp. 881–885.
- [10] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. Dimakis, "Finite length analysis of caching-aided coded multicasting," in *IEEE Allerton conference*, 2014.
- [11] Cisco, "The Zettabyte Era-Trends and Analysis," 2013.
- [12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM'99. Proceedings*. IEEE, 1999, vol. 1, pp. 126–134.
- [13] Z. Bar-Yossef, Y. Birk, T.S. Jayram, and T. Kol, "Index coding with side information," *Information Theory, IEEE Transactions on*, vol. 57, no. 3, pp. 1479–1494, 2011.
- [14] M. Garey and D. Johnson, *Computers and Intractability: a guide to the theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.
- [15] M. Garey and D. Johnson, "The Complexity of Near-Optimal Coloring," *Journal of the ACM*, vol. 23, pp. 43–49, 1976.
- [16] T.A. Feo and M.G.C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.
- [17] P. Festa and M.G.C. Resende, "GRASP: An annotated bibliography," in *Essays and Surveys on Metaheuristics*, C.C. Ribeiro and P. Hansen, Eds., pp. 325–367. Kluwer Academic Publishers, 2002.
- [18] P. Festa and M.G.C. Resende, "An annotated bibliography of GRASP – Part I: Algorithms," *International Transactions in Operational Research*, vol. 16, no. 1, pp. 1–24, 2009.
- [19] P. Festa and M.G.C. Resende, "An annotated bibliography of GRASP – Part II: Applications," *International Transactions in Operational Research*, vol. 16, no. 2, pp. 131–172, 2009.
- [20] P. Festa and M.G.C. Resende, "GRASP: Basic components and enhancements," *Telecommunication Systems*, vol. 46, no. 3, pp. 253–271, 2011.
- [21] M. Laguna and R. Martí, "A GRASP for coloring sparse graphs," *Computational Optimization and Applications*, vol. 19, no. 2, pp. 165–178, 2001.

⁶Note that LFU may be slightly better than GRASP only for small B and very small M (negligible caching benefit), as shown in Fig. 7(a).